Structack: Structure-based Adversarial Attacks on Graph Neural Networks

Hussain Hussain
hussain@tugraz.at
Graz University of Technology
Austria
Know Center GmbH
Austria

Tomislav Duricic
tduricic@tugraz.at
Graz University of Technology
Austria
Know Center GmbH
Austria

Elisabeth Lex
elisabeth.lex@tugraz.at
Graz University of Technology
Austria

Denis Helic
dhelic@tugraz.at
Graz University of Technology
Austria

Markus Strohmaier
markus.strohmaier@cssh.rwth-aachen.de
RWTH Aachen
Germany
GESIS - Leibniz Institute for the Social Sciences
Germany

Roman Kern
rkern@tugraz.at
Graz University of Technology
Austria
Know Center GmbH
Austria

ABSTRACT

Recent work has shown that graph neural networks (GNNs) are vulnerable to adversarial attacks on graph data. Common attack approaches are typically *informed*, i.e. they have access to information about node attributes such as labels and feature vectors. In this work, we study adversarial attacks that are *uninformed*, where an attacker only has access to the graph structure, but no information about node attributes. Here the attacker aims to exploit structural knowledge and assumptions, which GNN models make about graph data. In particular, literature has shown that structural node centrality and similarity have a strong influence on learning with GNNs. Therefore, we study the impact of centrality and similarity on adversarial attacks on GNNs. We demonstrate that attackers can exploit this information to decrease the performance of GNNs by focusing on injecting links between nodes of low similarity and, surprisingly, low centrality. We show that structure-based uninformed attacks can approach the performance of informed attacks, while being computationally more efficient. With our paper, we present a new attack strategy on GNNs that we refer to as Structack. Structack can successfully manipulate the performance of GNNs with very limited information while operating under tight computational constraints. Our work contributes towards building more robust machine learning approaches on graphs.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Adversarial learning**; **Neural networks**.

HT '21, August 30-September 2, 2021, Virtual Event, Ireland

ACM ISBN 978-1-4503-8551-0/21/08.
https://doi.org/10.1145/3465336.3475110

KEYWORDS

Graph neural networks; adversarial attacks; network centrality; network similarity

ACM Reference Format:
Hussain Hussain, Tomislav Duricic, Elisabeth Lex, Denis Helic, Markus Strohmaier, and Roman Kern. 2021. Structack: Structure-based Adversarial Attacks on Graph Neural Networks. In *Proceedings of the 32nd ACM Conference on Hypertext and Social Media (HT '21), August 30-September 2, 2021, Virtual Event, Ireland*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3465336.3475110

1 INTRODUCTION

Graph neural networks (GNNs) are state-of-the-art models for tasks on graphs such as node classification [13], link prediction [31] and graph classification [9]. Recent work has shown that GNNs are vulnerable to adversarial attacks, which can cause GNNs to fail by carefully manipulating node attributes [16], graph structure [17, 34] or both [33]. For example, adversarial attacks on social networks can add links via fake accounts, or change the personal data of a controlled account. Most existing attacks [6, 16, 17, 30, 33, 34] assume that information about node attributes (e.g., demographics of users) are available to the attacker. In practice however, attackers have limited access to such attribute information. We thus differentiate between two cases: the *informed* case where both graph structure and node attributes are available to the attacker, and the *uninformed* case where only information about the structure is available (see Figure 1).

Objectives. In this work, we investigate *uninformed* adversarial attacks that aim to reduce the overall accuracy of node classification with GNNs by manipulating the graph structure. Our aim is to study (i) potential strategies for uninformed attacks and (ii) how effective they are in practical settings.

Approach. Insights in [12, 16, 34] have shown a considerable influence of node degree and shortest paths on GNN robustness.

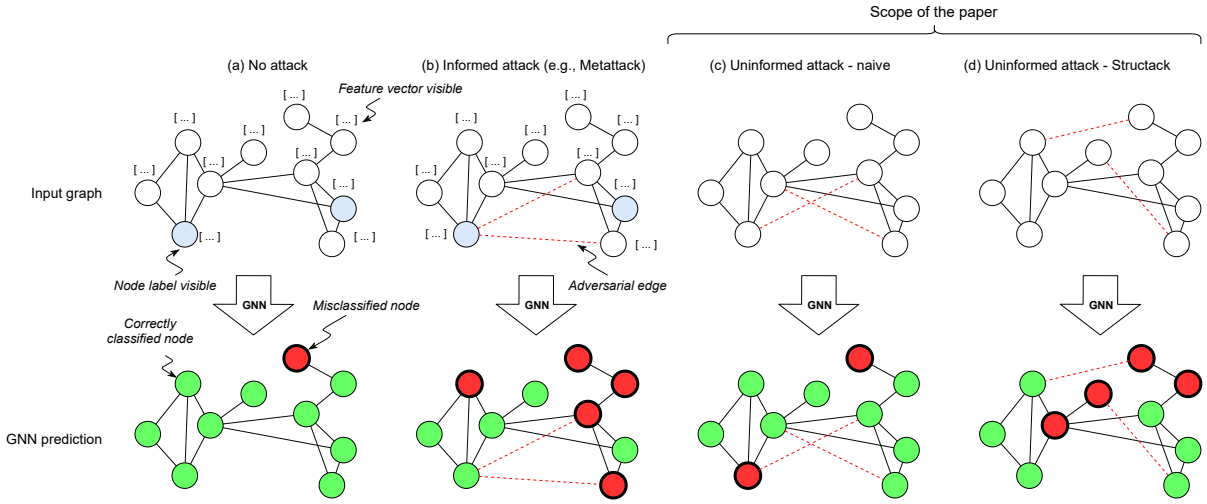

Figure 1: Illustration of Structack on GNN classification. In (a) we show a standard GNN classification with all node features and labels available to the algorithm. In (b) we depict an informed attack, which has access to the same information that the GNN classification task itself has. Based on this information, adversarial edges are added to attack GNN performance. In (c), we show an uninformed attack strategy, i.e. an attack that has no access to information about node attributes (labels, features), but only to the structure of the graph. A naive strategy could add edges based on topological graph features, for example add edges between pairs of nodes with high centrality or high similarity. In (d), we show a Structack attack which also has no access to information about node attributes, but attacks more successfully by adding edges between nodes with low centrality and low similarity. Structack approaches the performance of informed attacks such as Metattack [34] with less available information.

However, these insights are not well investigated. Therefore, we further inspect the effect of degree centrality and shortest path lengths on GNN adversarial attacks. First, we theoretically show that with standard degree normalization, low-degree neighbors surprisingly have more influence on a node's representation than higher-degree neighbors. Second, we discuss the results showing the dependency of GNNs on links within graph communities [10, 14], which are ubiquitous in real-world graphs. Based on that, we argue that adversarial edges should link nodes with longer paths between them. Experimentally, we verify these insights on degrees and distance through simulating attacks on empirical datasets. We then introduce our uninformed **struc**ture-based adversarial at**tack** (Structack), which generalizes these findings, and injects links between nodes of low structural centrality and similarity. Finally, we evaluate Structack compared to state-of-the-art attacks in terms of (i) reducing GNN accuracy, (ii) computational efficiency, and (iii) the ability to remain undetected.

Contribution and Impact. We introduce Structack[1], a novel structure-based uninformed adversarial attack on GNNs. In experiments on empirical datasets, Structack performs on a level that is comparable to more informed state-of-the-art attacks [30, 34], while using less information about the graph and significantly lower computational requirements. We give insights on the detection of attacks, such as Structack, by analyzing their ability to be undetected. With our work, we introduce a new unstudied category of attacks that could be applied to real-world networks. Our findings highlight the vulnerability of GNNs to uninformed attacks that have no knowledge about node attributes or the attacked model. Hence, our work contributes toward building more robust predictive and defensive models for graph data.

[1]We provide the implementation Structack and the experiments for reproducibility at https://github.com/sqrhussain/structack.

2 BACKGROUND

Preliminaries. Let $G = (A, X, Y)$ be an attributed undirected graph with an unweighted adjacency matrix $A \in \{0,1\}^{n \times n}$, a feature matrix $X \in \mathbb{R}^{n \times f}$, and a label matrix $Y \in \{0,1\}^{n \times |L|}$, where L is the set of labels. We refer to the set of nodes as $V = \{1, ..., n\}$, and the set of edges as E, where $|E| = m$ and $(i, j) \in E$ *iff* $A_{i,j} = 1$. Each node $u \in V$ has a feature vector $x_u \in \mathbb{R}^f$, where f is the feature vector dimension, and a label $y_u \in L$. The feature vectors are encoded in X, where u's feature vector x_u^T is the row u of matrix X. The labels of all the nodes are accordingly encoded in Y as well, with one-hot encoding in each row. We use the notation D to refer to the degree matrix, a diagonal matrix where $D_{i,i} = d_i$ is the degree of node i.

Graph neural networks. GNNs are multi-layer machine learning models that process graph-structured data. They follow a message passing and aggregation scheme, where nodes aggregate the messages received from their neighbors and update their representation on this basis. For a GNN with K layers, Kipf and Welling [13] describe the propagation rule in a simple form as follows

$$H'^{(k+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H'^{(k)}W^{(k)}) \quad (1)$$

for $k = 0, 1, ..., K-1$, where $\tilde{A} = A + I$, $\tilde{D} = D + I$, $H'^{(0)} = X$ are the given node features, $W^{(k)}$ is a trainable weight matrix, and σ

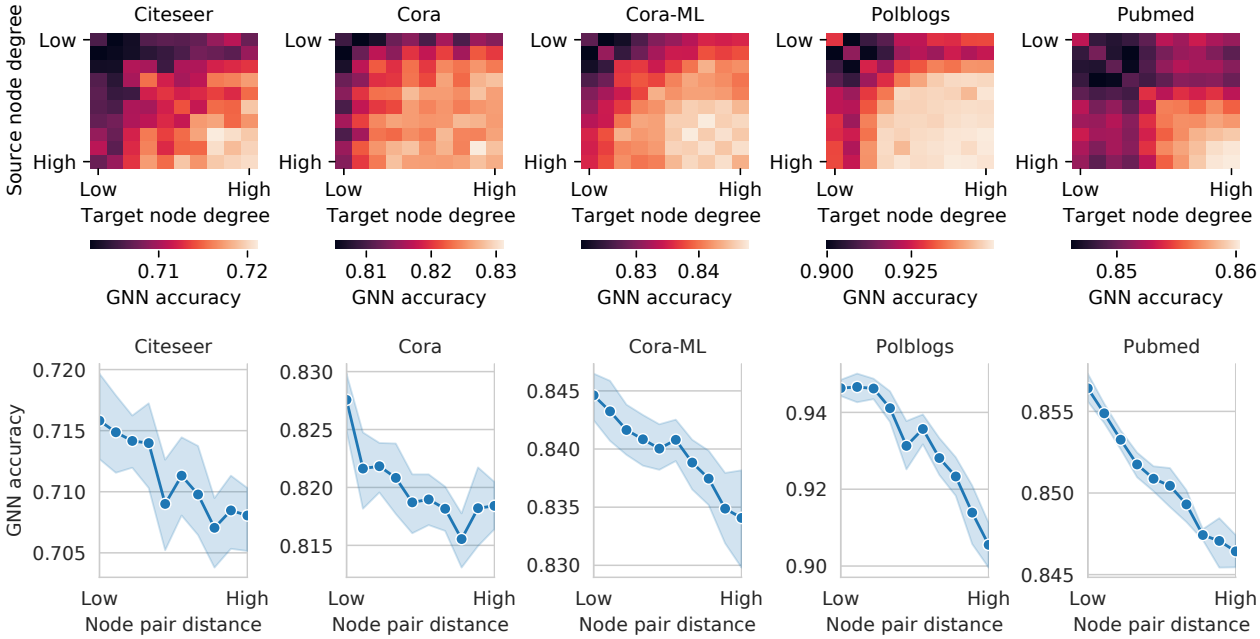

Figure 2: Impact of degree and distance on adversarial attacks on GNN classification. *Top*: GNN accuracy when we link nodes of varying degrees to other nodes of varying degrees as well, i.e., low-to-low degrees (top-left corner) up to high-to-high (bottom-right corner). Linking nodes with lower node degrees appears to result in more effective attacks. *Bottom*: GNN accuracy when adding edges between pairs of nodes with the lowest distance up to the highest distance. Linking nodes with higher distance (lower similarity) results in a more effective attack. Degrees and distances are grouped into 10-quantiles. The presented accuracy comes from training a GNN model (namely GCN[13]) on the perturbed graphs of 5 empirical datasets.

is an activation function, which is typically non-linear, e.g., ReLU. This formula is usually written as $H'^{(k+1)} = \sigma(\hat{A}H'^{(k)}W^{(k)})$ with $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$.

Node classification with GNNs. For node classification, we set the activation function of the last layer of the GNN to softmax

$$Z = f(\Theta; A, X) = \text{softmax}(\hat{A}H'^{(K-1)}W^{(K-1)}), \tag{2}$$

where $\Theta = \{W^{(0)}, ..., W^{(K-1)}\}$ is the set of model parameters which we aim to optimize, and $Z_{u,c}$ represents the model confidence that a node u belongs to label c. Given the labels of a subset of nodes $V' \subseteq V$, the goal of node classification is to find the labels of the unlabeled nodes $V \setminus V'$. To achieve this with GNNs, a common choice is to minimize the cross entropy error in V'

$$\mathcal{L}(Y, Z) = -\sum_{u \in V'} \ln Z_{u,y_u}, \tag{3}$$

where Z_{u,y_u} represents the model confidence that node u belongs to its ground-truth class y_u.

Adversarial attacks on GNNs. GNNs are prone to global adversarial attacks on the graph structure (i.e., the adjacency matrix A) [11]. These attacks aim to reduce the overall (i.e., *global*) node classification accuracy of a GNN model. To that end, these attacks follow different strategies to perturb the graph structure by adding or removing up to a *budget* of k edges.

3 IMPACT OF THE STRUCTURE ON ATTACKS

This section presents examples of graph structural properties and analyzes their effect on adversarial attacks on graph structure. Findings from related work show that node degrees have an impact on graph controllability [15] and on the selection of targeted nodes in adversarial attacks on node attributes [16]. Besides, an analysis of Metattack (a state-of-the-art attack) in [34] shows a slight tendency of the attack to link pairs of nodes with longer shortest paths (i.e., longer distances[2]). However, these works do not particularly focus on the impact of node degrees and distances on adversarial attacks or the reasoning behind it. Therefore, in the following analysis, we study the impact of node degrees and distances on GNNs from a theoretical perspective, and consequently verify this impact empirically.

3.1 Impact of degree and distance

Node degree impact. We aim to theoretically assess the role of node degree on the propagation in GNNs (Equation 1). For this study, we investigate the common degree normalization form as in Equation 1, i.e., normalization by the degree square root of two adjacent nodes. Using other less common forms of degree normalization or no degree normalization can be investigated in future work. We first simplify the update rule given in Equation 1 by ignoring the non-linearity in the intermediate layers, i.e., linearizing

[2]We refer to shortest path lengths as distances for brevity.

the equation (inspired by [33] and [28])

$$H'^{(K)} := \text{softmax}(H^{(K)}W) = \text{softmax}(\hat{A}^K XW), \tag{4}$$

where weight matrices $W^{(k)}$ for $k \in \{0, 1, .., K-1\}$ are absorbed by $W = W^{(0)}W^{(1)}...W^{(K-1)} \in \mathbb{R}^{f \times |L|}$. We use $H^{(k)} = \hat{A}^k X \in \mathbb{R}^{n \times f}$ to represent node intermediate representations at layer k in the linearized model. Each row u of matrix $H^{(k)}$, denoted as $(h_u^{(k)})^T \in \mathbb{R}^f$, is the intermediate representation of node u at layer k. As $H^{(k)} = \hat{A}H^{(k-1)}$, we can write the representation in layer k of node u (i.e., $h_u^{(k)}$) in terms of the representations of its neighboring nodes $\mathcal{N}(u)$ in the previous layer $k-1$ as follows

$$h_u^{(k)} = \sum_{v \in \mathcal{N}(u)} \frac{1}{\sqrt{d_u d_v}} h_v^{(k-1)}. \tag{5}$$

To show the impact of the degree of a specific neighbor $w \in \mathcal{N}(u)$ on the node u, we compute the derivative of u's final representation $h_u^{(K)}$ with respect to w's initial representation $h_w^{(0)}$ (i.e., the input features for node w), that is, the Jacobian matrix $\mathbf{J}^{u,w} \in \mathbb{R}^{f \times f}$ with $J_{i,j}^{u,w} = \partial h_{u,i}^{(K)} / \partial h_{w,j}^{(0)}$. Equation 5 shows that the i-th vector component of $h_u^{(k)} : k > 0$ (i.e., $h_{u,i}^{(k)}$) only depends on the vector component $h_{w,i}^{(k-1)}$ of the neighbor w, and not on any other component $h_{w,j}^{(k-1)}$ with $i \neq j$[3]. By induction, we can show that, for $w \in \mathcal{N}(u)$, the i-th vector component of $h_u^{(k)}$ only depends on the i-th vector component of $h_w^{(0)}$. This fact leads to the Jacobian matrix being diagonal. Therefore, it is sufficient to compute the partial of an arbitrary component i

$$J_{i,i}^{u,w} = \frac{\partial h_{u,i}^{(K)}}{\partial h_{w,i}^{(0)}} = \frac{\partial (\sum_{v \in \mathcal{N}(u)} \frac{1}{\sqrt{d_u d_v}} h_v^{(k-1)})}{\partial h_{w,i}^{(0)}}. \tag{6}$$

By applying the chain rule, we get

$$J_{i,i}^{u,w} = \sum_{v_1 \in \mathcal{N}(u)} \frac{1}{\sqrt{d_u d_{v_1}}} \frac{\partial h_{v_1,i}^{(K-1)}}{\partial h_{w,i}^{(0)}} \tag{7}$$

By repeatedly applying the chain rule K times, we end up at the partial of a node's initial representation $h_{v_K,i}^{(0)}$ in terms of w's initial representation $h_{w,i}^{(0)}$, that is

$$\frac{\partial h_{v_K,i}^{(0)}}{\partial h_{w,i}^{(0)}} = \begin{cases} 1 : v_K = w \\ 0 : v_K \neq w. \end{cases} \tag{8}$$

When we propagate this back to Equation 7, we arrive at

$$J_{i,i}^{u,w} = \sum_{v_1 \in \mathcal{N}(u)} \frac{1}{\sqrt{d_u d_{v_1}}} (...(\sum_{v_K \in \{w\}} \frac{1}{\sqrt{d_{v_{K-1}} d_{v_K}}})...) \tag{9}$$

We can rewrite Equation 9 for each (not necessarily simple) path of length K between u and w, that is, with $K-1$ intermediate nodes $[v_1, v_2, ..., v_{K-1}] \in \texttt{Paths}(u, w, K)$, as follows

$$J_{i,i}^{u,w} = \frac{1}{\sqrt{d_u d_w}} \sum_{[v_1, v_2, ..., v_{K-1}] \in \texttt{Paths}(u,w,K)} \prod_{i=1}^{K-1} \frac{1}{d_{v_i}} \tag{10}$$

[3]This argument is possible due to linearizing the Equation 1.

Table 1: Dataset statistics.

Dataset	Nodes	Edges	Features	Labels
Citeseer [24]	2,110	3,668	3,703	6
Cora [24]	2,485	5,069	1,433	7
Cora-ML [18]	2,810	7,981	2,879	7
Polblogs [1]	1,222	16,714	1,490	2
Pubmed [19]	19,717	44,325	500	3

This final term is $O((d_u d_w)^{-1/2})$ in terms of the two neighbors degrees. This shows that high-degree nodes have less impact on the representations of their neighbors, and that the degree normalization is the main reason. While the normalization is essential to reduce the bias towards nodes with very high degree, e.g., hubs, it can also make GNNs more vulnerable to attacks from nodes with low degrees.

Node distance impact. To explain the impact of node distance on attacks, we start by discussing the relationship between GNNs and network communities. Then we infer the role of the distance in this context. Communities are densely connected subgraphs, and are very common in empirical networks, e.g., social networks and co-author networks. The GNN update rule (Equation 1) is a special form of Laplacian smoothing [14]. Crucially, GNNs assume that nodes within the same community tend to share the same label and have similar features [10, 14]. This indicates that linking nodes from different communities effectively perturbs the GNN accuracy.

Findings in [3] show that nodes within similar communities tend to have shorter paths between them. This supports that linking nearby nodes is likely adding intra-community links, while linking more remote nodes is likely adding inter-community links. Therefore, we hypothesize that linking distant nodes would result in more effective attacks.

3.2 Empirical validation

Next, we empirically verify the hypotheses from the previous analysis on the datasets, summarized in Table 1. We perform perturbation by adding edges to the graph following different strategies. Then we observe the accuracy of training a GNN model on the perturbed graph. We choose the well-known non-linear GCN [13] model[4] to empirically show that our theoretical analysis of a linearized GNN model extends to a non-linear one. In the next experiments, we have a budget of $k = \lfloor r \times m \rfloor$ edges to add to the graph, where r is the perturbation rate which we set to 0.05.

Node degree. The first experiment aims to compare linking low-degree nodes to linking high-degree nodes. We group the nodes into 10 equal-sized subsets based on their degrees. For each pair of subsets, we try adding k adversarial edges between random pairs of nodes in the two subsets and observe the GCN accuracy. We obtain the results in Figure 2 (top). These results support our discussion (Section 3.1) and show an increase in accuracy, i.e., a decrease in attack effectiveness, when linking pairs of high-degree nodes. As a

[4]As the reader might notice, the analysis in Section 3.1 does not only apply to this particular family of GNNs since feature propagation and normalization are necessary components of GNNs. Our work studies SGC models theoretically and GCN models empirically.

result, we assume that attacks are more effective when they *link pairs of low-degree nodes*.

Node distance. The second experiment aims to compare linking distant pairs of nodes to linking nearby pairs. We perform this experiment in 10 trials, with trial 1 linking nodes with lowest distances and trial 10 with highest distances. In each trial, we observe the GCN accuracy after adding k adversarial edges. In trial $i \in \{1, .., 10\}$, for each adversarial edge (to be added), we randomly pick one node u from the graph and attach one end of that edge to u. Then, we group all the nodes in the graph into 10 equal-sized subsets based on their distance from u. Finally, we link u to a random node in the i-th subset. Figure 2 (bottom) depicts this comparison and shows the accuracy of each trial. The figure suggests that *linking distant nodes results in more effective attacks* than linking nearby nodes.

4 STRUCTACK

In this section, we introduce our attack strategy Structack (**Struc**ture-based at**tack**), built upon the findings from Section 3. We outline the attacker's goal, capabilities and knowledge, explain the attack strategy, provide a complexity analysis, and discuss insights on the detection of the attack.

4.1 Attacker's capabilities and restrictions

In our setting, the attacker aims to minimize the overall GNN accuracy on node classification. We limit the knowledge of the attacker to the adjacency matrix, as opposed to existing work [6, 16, 17, 30, 33, 34]. The attacker has no access to the features or the label of any node. They also do not have any information about the attacked GNN model or its parameters. We assume that the attacker is able to add edges between any pair of nodes[5] in the graph, up to a limit k, called the **budget**. As a result, the attack generates a poisoned adjacency matrix A', where $||A - A'||_0 \leq k$. According to the taxonomy suggested by [11], our attack is an untargeted (global) poisoning attack on graph structure.

4.2 Attack strategy

The findings in Section 3 show the impact of low node degrees and long node distances in the graph on adversarial attacks. Following these findings, an efficient strategy to exploit this impact is to (1) *select* nodes with low degrees, and (2) *link* pairs of nodes with high distances. Node degree is a measure of node centrality, and distance represents one form of node dissimilarity (e.g., Katz similarity [20] gives higher weights to shorter paths). We generalize node degree and distance to a diverse set of measures of centrality and similarity. Therefore, Structack consists of selecting nodes with the lowest *centrality* and linking these nodes so that the *similarity* between linked nodes is minimized.

For a budget k, Structack chooses $2k$ nodes with the lowest centrality. We then split these nodes into two sets U_1 and U_2, both of size k, based on their centrality, i.e., U_1 has the k nodes with lowest centrality. Then Structack finds the matching between nodes in U_1 and U_2, which minimizes the sum of similarities between the matched nodes. To solve this minimization problem, we use the Hungarian algorithm. Finally, Structack adds edges between matched nodes.

For selection and linking steps, we investigate different choices of centrality and similarity measures (Table 2). Otherwise, we follow conventional procedures, e.g., splitting lowest-centrality nodes in order, and using the sum of similarities as a criterion for the matching problem. Please note, investigating other splitting and matching criteria can be interesting, e.g., using interleaving splitting. However, we leave this for future work as we are more interested in the impact of centrality and similarity choices.

4.3 Complexity analysis

After computing the centrality for each node, obtaining the $2k$ lowest-centrality nodes for the splitting step requires $O(n \log k)$ time. At the final step of Structack, finding the optimal node matching is a minimum cost maximum bipartite matching problem. We solve this problem using the Hungarian algorithm, which has the complexity of $O(k^3)$ time.

In Table 2, we list the centrality and similarity measures we used with their corresponding time and memory complexity. These measures are well defined in the literature, along with their complexity. However, to make our paper self-contained, we explain essential details about how we compute similarity and give the resulting time complexity.

Community-based similarity: First, we perform community detection using Louvain method [4], which splits the graph into C disjoint communities. We then build a community similarity matrix $\mathbf{S} \in \mathbb{R}^{C \times C}$ encoding the original density of edges, i.e., $\mathbf{S}_{i,j}$ represents the edge density of links between community i and community j. Then we set the similarity between two nodes u and v to the similarity of their corresponding communities $\mathbf{S}_{Comm(u),Comm(v)}$, where $Comm(x)$ is the community of node x as per Louvain method. For the community-based similarity, the time complexity of Louvain community detection is considered to be linear in the number of edges on typical and sparse data [4] $O(m)$, and the edge density computation step is also of order $O(m)$, making this similarity calculation of order $O(m)$ as well.

Distance-based similarity: We use breadth-first search (BFS) to get single-source shortest paths from each node in U_1 to all nodes in U_2 (which, in the worst case, means to all nodes in the graph). We choose BFS because we assume that the input graph is unweighted as mentioned in Section 2. We restrict BFS sources to nodes in U_1 since the distance between nodes outside U_1 and U_2 are not relevant for Structack. For the shortest path length computation, and if we do not consider parallelization, the BFS algorithm is repeated k times (once for each node in U_1), which gives a time complexity of $O(km)$. Please note that a higher distance indicates a lower similarity.

Katz similarity: This notion is a measure of regular equivalence of nodes [20] . It counts paths of all lengths and weighs them differently, i.e., shorter paths with higher weights. We can write Katz similarity matrix as $\sum_{i=0}^{\infty} (\alpha A)^i$, where α is a constant which needs to be less than the inverse of the largest eigenvalue of A. We approximate the similarity matrix without matrix inversion using *inverse iteration* until the matrix converges after t iterations. With sparse matrix multiplication, the time complexity turns into $O(tm)$.

[5] This ability might not directly translate to real-world attacks, but it is necessary to study the extent of different attack approaches, including the baselines that we evaluate as well.

Table 2: Description of considered centrality/similarity metrics with their time and memory complexity. We list abbreviations for the metric names to use later in results tables. n and m represent the number of nodes and edges in the graph respectively, and k is the attack budget. t is the number of iterations for computing Pagerank centrality and Katz similarity.

Centrality metric	Time complexity	Memory complexity
Degree (DG)	$O(m)$	$O(m)$
Eigenvector (EV) [21]	$O(m)$	$O(n+m)$
Pagerank (PR) [22]	$O(tm)$	$O(n+m)$
Betweenness (BT) [5, 21]	$O(nm)$	$O(n+m)$
Closeness (CL) [7, 21]	$O(nm)$	$O(n+m)$
Similarity metric	**Time complexity**	**Memory complexity**
Katz (Katz) [20]	$O(tm)$	$O(n^2)$
Community-based (Comm)	$O(m)$	$O(m)$
Distance-based (Dist)	$O(km)$	$O(m)$

The number of iterations goes down to the desired precision of the similarity. For a typical choice of $\alpha = 0.85$, we obtain a precision of 10^{-6} with $t = 100$ iterations[6].

4.4 Insights on attack detection

Structack selects nodes with low centrality, which typically have few edges. Therefore, the attack can cause a significant change in the degree distribution. We hence suggest observing the changes in the degree distribution, similar to [33].

Moreover, Structack links pairs of nodes with low structural similarity, which are likely to have few common neighbors. The local clustering coefficient of a node is lower with fewer edges shared among its neighbors [20]. Based on that, we expect that Structack causes a significant change in the local clustering coefficients of the nodes being linked. Therefore, we also suggest observing the changes in the local clustering coefficient distribution.

We define two criteria to detect the attack by comparing the original and the perturbed graphs. First, we test if the node degrees of both graphs stem from the same distribution. Second, we test if the local clustering coefficient values of both graphs also stem from the same distribution. If values are assumed to stem from the same distribution in both cases, we consider the attack to be *unnoticeable*.

5 EXPERIMENTAL EVALUATION

5.1 Adversarial attack evaluation

The goal of the experimental evaluation is to test the efficacy of Structack perturbations on GNNs. To this end, we evaluate Structack against informed baseline attacks as well as the random (uninformed) baseline. Notice that our attacks as well as the evaluated baselines apply structural perturbations only and not feature perturbations. For a perturbation rate r, we allow each attack to perturb the graph by adding (or removing in case of some studied baselines) a budget of $k = \lfloor r \times m \rfloor$ edges. We evaluate each attack on three different criteria: (i) Effectiveness in terms of GNN misclassification rate, (ii) Efficiency in terms of computation time and memory requirements, and (iii) Unnoticeability in terms of changes of degree and clustering coefficient distributions. With this evaluation, we aim to demonstrate a performance trade-off of these three aspects.

[6]This argument also applies for computing Pagerank which is also $O(tm)$.

5.2 Experimental setup

We evaluate 24 different combinations of (Structack) derived from combining 6 different possibilities for node selection (including random selection) with 4 different possibilities for node linking (including random linking) as listed in Table 2. We include random selection and random linking to evaluate whether the effectiveness of certain centrality or similarity choices stem from randomness. We perform the following evaluations on the 5 datasets described in Table 1.

Effectiveness. To evaluate effectiveness (misclassification), we train a GNN model on the perturbed graph and report the classification accuracy on its test set. Aiming for more robust evaluation (inspired by [25]), we use 5 different random splits (10% train, 10% validation, and 80% test) for each dataset. Our GNN model of choice is the well-known GCN [13] model, which we initialize 5 times with different random weights for each perturbed input graph. For the effectiveness evaluation, we set the perturbation rate to 0.05.

Efficiency. Another criterion for evaluating adversarial attacks is their ability to efficiently use available resources in terms of computation time and used memory. More efficient attacks have a lower runtime and use less memory. Please note that we ran all experiments on a machine running Linux Ubuntu OS version 16.04 with Intel Xeon E5-2630 Processor with 40 CPUs, 256GB RAM, and a dedicated NVIDIA Tesla P100 16GB GPU. For these efficiency experiments, we also set the perturbation rate to 0.05. If an attack did not fit into the GPU memory for a particular dataset, we ran it with CPU settings for that dataset.

Unnoticeability. To evaluate attack unnoticeability, we run each attack for different perturbation rates $r \in \{0.001, 0.002, 0.003, 0.004, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.10, 0.15, 0.20\}$. We report results in terms of the critical perturbation rate $r_{critical}$, i.e., largest r for which the attack is still deemed *unnoticeable*. We consider the attack to be unnoticeable if the changes in the node degree and local clustering coefficient values made by the attack are not significant. A commonly used approach for comparing two node degree distributions is the Two-Sample Kolmogorov-Smirnov statistical test (KS test) [2]. Therefore, we use the KS test to determine whether two compared samples (original graph versus perturbed graph) stem from the same distribution. We apply this test to obtain the significance in change for both degree and local clustering coefficient distributions. Here the null hypothesis of the KS test is that *two samples are drawn from the same continuous distribution*. We set the probability of rejecting the null hypothesis α to 0.05.

Baselines. We evaluate the most effective combinations of Structack against the following baselines in terms of the three evaluation criteria.

Random: A simple uninformed baseline attack that selects random node pairs and adds an edge between them. This is the only *uninformed* baseline against which we compare Structack.

DICE [27]: A simple heuristic, which is explicitly based on disconnecting nodes with the same label and connecting nodes

with different labels. This attack is *informed* as it has access to node labels.

Metattack [34]: State-of-the-art optimization-based attack on graphs via meta-learning. It treats the adjacency matrix as a parameter of the optimization problem, which is minimizing the accuracy of a surrogate model. Metattack does not require access to the GNN model parameters, and uses the surrogate model instead.

PGD and MinMax [30]: State-of-the-art optimization-based attacks on graphs. Both attacks apply projected gradient descent to solve the optimization problem after convex relaxation. MinMax attempts to build a more robust attack through attacking a retrainable GNN. These two attacks require access to the GNN model parameters.

In addition to the graph structure, Metattack, PGD, and MinMax have access to the feature vectors of all nodes and the labels of some nodes (typically, nodes in the training set). Thus, these three attacks are *informed* in our definition. These attacks involve randomization, which is why we initialize each of them 5 times with different random weights for each attack setting.

6 RESULTS AND DISCUSSION

Next we present evaluation results, discuss trade-offs, and outline the limitations of our work. In the results tables, we use the abbreviations defined in Table 2 to describe the centrality and similarity measures of Structack combinations.

Effectiveness. First, we apply Structack combinations to each graph dataset and obtain the GCN accuracy. Then we compute the average rank of each combination in terms of classification accuracy. We visualize the ranking in Figure 3 with a critical difference diagram. The thick horizontal bars in this figure group together the combinations with no significant difference[7] in ranks between them. The six lowest-ranked combinations (which involve randomness) perform significantly worse than the rest. This confirms that the improvement of Structack does not stem from randomness. We observe that the seven most effective combinations are not significantly different. Among these combinations, we frequently see Pagerank centrality, degree centrality and Katz similarity, which implies the effectiveness of these three measures. Node centrality in Structack has a substantial impact on effectiveness, relative to the node similarity. For example, performing selection with degree or Pagerank centrality and linking at random (Degree.Random and Pagerank.Random in Figure 3) seems to perform better than some combinations that do not involve random linking.

As the seven most effective combinations do not differ significantly from each other, we consequently compare them to the baselines as presented in Table 3. Structack combinations show a comparable performance to state-of-the-art methods, although they have no access to node attributes.

Efficiency. In Tables 4 and 5, we respectively show the runtime and the memory consumption of our most effective Structack combinations and existing adversarial attack methods. We notice a significant drop in runtime and memory consumption for Structack compared to the optimization-based attacks (Metattack, PGD, and MinMax). These three attacks did not fit in the available GPU memory for Pubmed, and therefore we ran them with CPU settings for this dataset. For Structack combinations, the similarity measure generally has a substantial effect on runtime and memory consumption, with community-based similarity being the most efficient. An exception to this rule is the runtime of Betweenness and Closeness centralities. For example on Pubmed, Betweenness and Closeness computation takes 325 and 66 minutes respectively, while the computation of Katz similarity takes 100 minutes. The time complexity of computing these two measures (Table 2) is $O(nm)$ making them impractically slow for large graphs.

Unnoticeability. We report the critical perturbation rate $r_{critical}$ for which the respective attack remains unnoticeable as per our definition in Sections 5.2. We present $r_{critical}$ for each approach in Table 6. For most datasets, Structack's $r_{critical}$ is on par or slightly lower than the informed approaches. We also observe that the

[7]For details on the computation of significance, we refer to the documentation of the R package `scmamp` https://cran.r-project.org/web/packages/scmamp/scmamp.pdf.

Table 3: Adversarial attack effectiveness. This table gives the accuracy of a GCN model trained on the perturbed graph generated by applying each adversarial attack (lower accuracy → more effective attack). Structack (in boldface) is comparable with the state-of-the-art attacks on most datasets with minimum knowledge. According to a Wilcoxon signed-rank test, each Structack approach is significantly more effective than the uninformed Random approach with $p < 0.01$ (after Bonferroni correction). * *Metattack could not run for Pubmed with 16GB GPU, and did not finish with CPU settings after 3 weeks running*

	Dataset	Citeseer	Cora	Cora-ML	Polblogs	Pubmed
	Clean	71.90±1.9	83.44±1.1	85.11±0.7	94.66±1.2	86.51±0.3
Informed	DICE	70.63±1.8	80.09±2.1	80.74±2.4	82.44±6.1	83.32±2.6
	Metattack	69.21±1.7	76.83±1.3	80.01±1.0	76.98±0.9	N/A*
	MinMax	68.95±0.8	78.45±1.1	83.39±0.5	85.02±2.2	84.97±0.6
	PGD	63.80±0.9	75.15±1.4	80.17±0.7	83.28±3.3	82.58±0.4
Uninformed	Random	72.64±1.2	80.56±0.5	82.24±0.6	83.57±3.4	85.90±0.3
	BT*Katz	71.83±1.0	78.77±0.5	80.27±0.7	76.41±1.8	84.38±0.2
	DG*Comm	71.89±0.9	78.51±0.6	80.12±0.6	75.65±1.3	84.79±0.3
	DG*Dist	71.66±1.0	78.80±0.5	80.15±0.5	78.87±2.0	84.55±0.3
	DG*Katz	71.33±1.1	78.98±0.5	80.17±0.6	76.27±1.4	84.34±0.3
	PR*Comm	71.67±1.0	78.85±0.5	80.51±0.6	75.25±1.4	84.54±0.4
	PR*Dist	71.38±1.0	78.53±0.5	80.19±0.6	78.09±2.2	84.20±0.3
	PR*Katz	71.67±1.0	78.40±0.5	80.06±0.7	75.99±1.6	84.08±0.3

Table 4: Runtime in minutes with 0.05 perturbation rate. Structack is in boldface.

	Dataset	Citeseer	Cora	Cora-ML	Polblogs	Pubmed
Informed	DICE	0.05	0.07	0.13	0.08	3.07
	Metattack	7.75	7.65	22.38	8.80	N/A
	MinMax	12.83	13.03	13.68	12.58	2,645.87
	PGD	12.15	12.08	12.35	11.10	1,569.55
Uninformed	**BT*Katz**	3.33	5.12	7.42	3.87	379.00
	DG*Comm	0.03	0.05	0.10	0.25	1.70
	DG*Dist	0.05	0.10	0.23	0.82	8.08
	DG*Katz	0.98	1.30	1.55	0.63	97.98
	PR*Comm	0.05	0.08	0.15	0.27	1.90
	PR*Dist	0.10	0.12	0.27	0.87	8.13
	PR*Katz	0.93	1.32	1.52	0.72	93.28

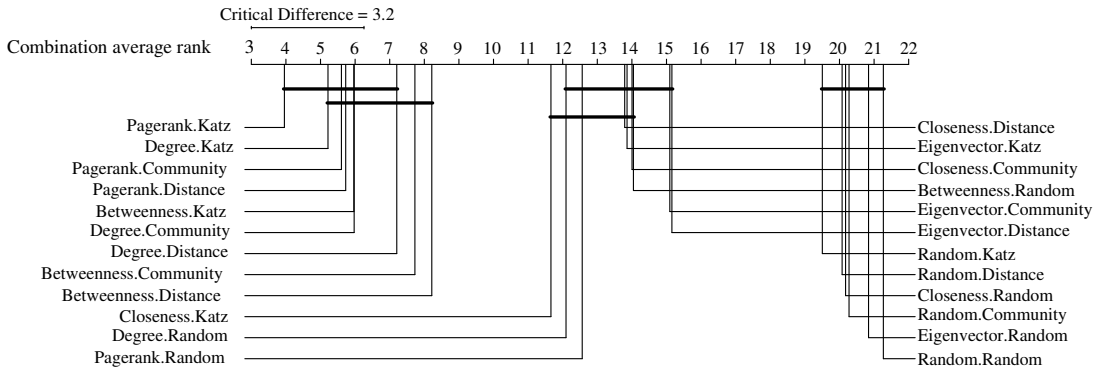

Figure 3: Comparison of Structack combinations' effectiveness. This plot shows combinations from most to least effective (lowest to highest GCN classification accuracy) presented from left to right. Thick horizontal bars represent no significant difference between the combinations they mark. We find that the best seven combinations are not significantly different, while being significantly better than the rest. We also see that the stronger impact lies in the choice of centrality with the degree and Pagerank centralities with random linking outperforming half of the other combinations.

Table 5: Memory consumption in Megabytes with 0.05 perturbation rate. Structack is in boldface. * *Snapshot taken after 3 weeks of running.*

	Dataset	Citeseer	Cora	Cora-ML	Polblogs	Pubmed
Informed	DICE	313	1,623	1,213	1,230	773
	Metattack	2,074	2,096	2,123	2,078	*58,394
	MinMax	2,176	2,243	2,318	2,109	20,554
	PGD	2,155	2,232	2,299	2,110	19,779
Uninformed	**BT*Katz**	578	626	677	442	13,918
	DG*Comm	316	322	337	403	901
	DG*Dist	433	431	430	433	1,995
	DG*Katz	556	587	662	440	13,918
	PR*Comm	445	443	443	460	928
	PR*Dist	445	443	442	450	2,021
	PR*Katz	570	617	668	441	13,919

Table 6: Maximum unnoticeable perturbation rate. We evaluate unnoticeability in terms of critical perturbation rate $r_{critical}$, which we define as as the maximum perturbation rate for which the attack remains unnoticeable as per definition in Section 5.2. We present the results for $r_{critical}$ per dataset and per adversarial attack. Structack is in boldface.

	Dataset	Citeseer	Cora	Cora-ML	Polblogs	Pubmed
Informed	DICE	0.0750	0.0500	0.0500	0.0100	0.0100
	Metattack	0.0100	0.0100	0.0100	0.0040	N/A
	MinMax	0.0750	0.0750	0.0750	0.0500	0.0040
	PGD	0.1000	0.0750	0.0500	0.1000	0.0040
Uninformed	Random	0.0250	0.0250	0.0250	0.0100	0.0050
	BT*Katz	0.0100	0.0100	0.0075	0.0020	0.0030
	DG*Comm	0.0100	0.0075	0.0050	0.0020	0.0030
	DG*Dist	0.0100	0.0075	0.0050	0.0020	0.0030
	DG*Katz	0.0100	0.0075	0.0050	0.0020	0.0030
	PR*Comm	0.0100	0.0075	0.0050	0.0020	0.0030
	PR*Dist	0.0100	0.0075	0.0050	0.0020	0.0030
	PR*Katz	0.0100	0.0075	0.0050	0.0020	0.0030

choice for node selection strategy in Structack has a greater influence on the attack unnoticeability than the node linking strategy.

Performance trade-off. Structack provides competitive effectiveness and high efficiency. However, it shows to be relatively noticeable compared to baseline approaches. On the other hand, optimization-based informed attacks achieve better unnoticeability but with much lower efficiency compared to Structack. This low efficiency prevents them from running on larger graphs, with Pubmed as a toy example (this has been recently noted by Geisler et al. [8]). A deeper look into Structack shows that the selection strategy (i.e., centrality measure) has more impact on effectiveness and unnoticeability. Conversely, the linking strategy (i.e., similarity measure) has more impact on the efficiency.

All in all, we assume an attack to be effective (cause high misclassification rate) if one of the 7 most effective combinations is picked. When running on big graphs, attackers would tend to choose efficient combinations such as DG×Comm. To hide their behavior, attackers would tend to choose less noticeable combinations such as BT×Katz.

Limitations. Our study focuses on exploiting the structure information using centrality and similarity measures. One could study other centrality and similarity measures, and even other graph structural properties. Moreover, instead of the theoretical strategy defined in Section 4.2, one could define a more practical heuristic to exploit these structural features. Furthermore, other forms of degree normalization in the target GNN model could result in different strategies than Structack, which is an interesting direction for future work. However, the aim of our work is to illustrate the extent to which uninformed attacks are successful, and we demonstrate that through our Structack strategy, which covers a range of possibilities of uninformed attacks.

Our unnoticeability measure was limited to degree and clustering coefficient distributions. Different unnoticeability tests could be investigated for this purpose. In this regard, Structack appears more noticeable than existing informed attacks due to its greediness in selecting nodes with lowest centrality. The unnoticeability results motivate us to look into approaches that intrinsically consider both effectiveness and unnoticeability. More careful selection

could improve Structack's unnoticeability, at the possible cost of effectiveness.

Additionally, comparing distributions of the clean graph and the perturbed one is not practical for dynamic networks, where edges and nodes are added and removed constantly. This comparison does not consider the natural growth of the network. This type of comparison is a common practice in works on adversarial attacks on graphs, and it should be improved. This could be alleviated by using graph growth models or dedicated datasets with edge timestamps.

7 RELATED WORK

Information available to attackers. Many recent works have introduced attack models for GNNs with different knowledge and capabilities. These models adhere to various restrictions on the practicality of the adversarial attacks and the limitations of the attacker. However, the majority of these models assume the attacker's knowledge of the targeted GNN model [29, 30] or their access to node attributes [16, 26, 33, 34], i.e., feature vectors and some labels. We have referred to such adversarial attacks as *informed attacks*. A recent survey [11] describes the level of knowledge of (i) the targeted GNN model and (ii) graph data as one characteristic of the attack. Our work differentiates between these two descriptions and focuses on the knowledge of graph data regardless of the knowledge of the targeted model.

Node centrality. Earlier findings in network science on controlling complex networks [15] show that fewer nodes are needed to control the network, if one aims to control nodes with low degrees. Another study about the stability of node embedding [23] shows that high-centrality nodes have more stable embeddings compared to low-centrality nodes. In the context of GNNs, Metattack[34] shows a slight tendency to connect nodes with low degree. Zhu et al. [32] experimentally consider attacks on nodes with higher than 10 degrees for noticeability considerations. Ma et al. [16] introduce practical adversarial attacks by targeting nodes with high importance score, e.g., PageRank, node degree, and betweenness. The authors argue that nodes with too high importance score, e.g., hubs, are hard to control, hence the attack approach avoids such nodes. Our work conversely builds theoretical grounds and experimental support to show that attacks are more effective if they focus on low degree nodes.

Node similarity. A study on the behavior of GNNs [14] shows that feature and label smoothness are the reason why GNNs work. Some works on GNN adversarial attacks [11, 12] analyze the poisoned graphs of popular attack models and show a tendency of the attackers to add edges between nodes with different labels and low-similarity features. Waniek et al. [27] introduce an attack that is explicitly based on disconnecting nodes with the same label and connecting nodes with different labels (Disconnect Internally, Connect Externally - DICE). More insights on structure in Metattack [34] suggest that attacks tend to link pairs of nodes with higher-than-average shortest path length. Finally, a preprocessing-based defense mechanism for GNNs [29] is based on reducing the weight of edges between nodes with a low Jaccard similarity score of their features. Our work builds on these findings to investigate more in structural node similarity and build an uninformed structure-based adversarial attack strategy.

8 CONCLUSION

We investigated the effectiveness of uninformed adversarial attacks on GNNs, i.e. attacks that have no access to information about node labels or feature vectors in a graph. With theoretical considerations and experimental support, we demonstrated that uninformed attacks can exploit structural features of the graph, such as node centrality and similarity. We presented Structack, a novel uninformed attack strategy that selects nodes with low centrality and links pairs of nodes with low similarity. In experiments on five graph datasets Structack showed comparable performance to state-of-the-art attacks, while having less information about the graph (no access to node attributes), exhibiting higher efficiency, and reasonable unnoticeability. Our work shows that uninformed adversarial attacks are successful with only structural knowledge, sometimes outperforming informed attacks. The feasibility of Structack on real-world graphs makes it vital to develop more structure-aware defense mechanisms for more reliable GNN prediction.

ACKNOWLEDGEMENTS

The Know-Center is funded within the Austrian COMET Program – Competence Centers for Excellent Technologies – under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG. This work is supported by the H2020 project TRUSTS (GA: 871481) and the "DDAI" COMET Module within the COMET Program, funded by the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit), the Austrian Federal Ministry for Digital and Economic Affairs (bmdw), the Austrian Research Promotion Agency (FFG), the province of Styria (SFG) and partners from industry and academia.

REFERENCES

[1] Lada A. Adamic and Natalie Glance. 2005. The Political Blogosphere and the 2004 U.S. Election: Divided They Blog. In *Proceedings of the 3rd International Workshop on Link Discovery* (Chicago, Illinois) *(LinkKDD '05)*. Association for Computing Machinery, New York, NY, USA, 36–43. https://doi.org/10.1145/1134271.1134277

[2] Sadegh Aliakbary, Jafar Habibi, and Ali Movaghar. 2014. Quantification and comparison of degree distributions in complex networks. In *7'th International Symposium on Telecommunications (IST'2014)*. 464–469. https://doi.org/10.1109/ISTEL.2014.7000748

[3] Sharmodeep Bhattacharyya and Peter J Bickel. 2014. Community detection in networks using graph distance. *arXiv preprint arXiv:1401.3915* (2014).

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[5] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology* 25, 2 (2001), 163–177.

[6] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371* (2018).

[7] Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks* 1, 3 (1978), 215–239.

[8] Simon Geisler, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Attacking Graph Neural Networks at Scale. In *Deep Learning for Graphs at AAAI Conference on Artificial Intelligence 2021, AAAI workshop 2021.*

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[10] Hussain Hussain, Tomislav Duricic, Elisabeth Lex, Roman Kern, and Denis Helic. 2020. On the Impact of Communities on Semi-supervised Classification Using

Graph Neural Networks. In *International Conference on Complex Networks and Their Applications*. Springer, 15–26.
[11] Wei Jin, Yaxin Li, Han Xu, Yiqi Wang, and Jiliang Tang. 2020. Adversarial Attacks and Defenses on Graphs: A Review and Empirical Study. *arXiv preprint arXiv:2003.00653* (2020).
[12] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. *Graph Structure Learning for Robust Graph Neural Networks*. Association for Computing Machinery, New York, NY, USA, 66–74. https://doi.org/10.1145/3394486.3403049
[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.
[14] Qimai Li, Zhichao Han, and Xiao Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. arXiv:1801.07606
[15] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. 2011. Controllability of complex networks. *nature* 473, 7346 (2011), 167–173.
[16] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. 2020. Black-Box Adversarial Attacks on Graph Neural Networks with Limited Node Access. *arXiv preprint arXiv:2006.05057* (2020).
[17] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. 2019. Attacking graph convolutional networks via rewiring. *arXiv preprint arXiv:1906.03750* (2019).
[18] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.
[19] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. 2012. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, Vol. 8.
[20] Mark Newman. 2018. *Networks*. Oxford university press.
[21] Mark EJ Newman. 2008. The mathematics of networks. *The new palgrave encyclopedia of economics* 2, 2008 (2008), 1–12.
[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.
[23] Tobias Schumacher, Hinrikus Wolf, Martin Ritzert, Florian Lemmerich, Jan Bachmann, Florian Frantzen, Max Klabunde, Martin Grohe, and Markus Strohmaier. 2020. The Effects of Randomness on the Stability of Node Embeddings. *arXiv preprint arXiv:2005.10039* (2020).
[24] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.
[25] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. *Relational Representation Learning Workshop, NeurIPS 2018* (2018).
[26] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. 2020. *Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach*. Association for Computing Machinery, New York, NY, USA, 673–683. https://doi.org/10.1145/3366423.3380149
[27] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour* 2, 2 (2018), 139–147.
[28] Felix Wu, Tianyi Zhang, Amaur Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. 2019. Simplifying graph convolutional networks. *Proceedings of Machine Learning Research* (2019).
[29] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 4816–4823.
[30] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3961–3967. https://doi.org/10.24963/ijcai.2019/550
[31] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*. 5165–5175.
[32] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1399–1407.
[33] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.
[34] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bylnx209YX

A DETAILED EFFECTIVENESS RESULTS

Table shows the accuracy of GCN [13] for node classification on the considered datasets after changing the structure using different combinations of Structack with a perturbation rate $r = 0.05$. These are the detailed results of what Figure 3 summarizes.

Table 7: GCN accuracy on each dataset after applying Structack with each centrality×similarity combination. The lowest accuracy (best combination) in each dataset is shown in boldface.

Dataset	Similarity Centrality	Community	Distance	Katz	Random
Citeseer	Betweenness	72.06±0.9	72.04±1.0	71.83±1.0	72.22±1.0
	Closeness	72.75±0.7	71.99±1.0	72.22±1.1	72.88±1.1
	Degree	71.89±0.9	71.66±1.0	**71.33±1.1**	71.86±0.9
	Eigenvector	72.44±1.1	72.55±0.8	72.46±1.3	73.05±0.8
	Pagerank	71.67±1.0	71.38±1.0	71.67±1.0	72.18±0.7
	Random	73.08±1.1	73.08±1.1	73.04±0.9	72.64±1.2
Cora	Betweenness	79.05±0.5	79.11±0.4	78.77±0.5	79.65±0.5
	Closeness	79.99±0.4	79.99±0.6	79.57±0.5	80.33±0.4
	Degree	78.51±0.6	78.80±0.5	78.98±0.5	79.63±0.5
	Eigenvector	80.19±0.5	79.80±0.5	79.93±0.6	80.53±0.5
	Pagerank	78.85±0.5	78.53±0.5	**78.40±0.5**	78.99±0.5
	Random	80.35±0.5	80.44±0.5	80.31±0.5	80.56±0.5
Cora-ML	Betweenness	80.50±0.7	80.16±0.5	80.27±0.7	80.85±0.7
	Closeness	81.42±0.7	81.32±0.7	81.58±0.6	82.14±0.6
	Degree	80.12±0.6	80.15±0.5	80.17±0.6	80.51±0.7
	Eigenvector	81.72±0.8	81.60±0.5	81.48±0.7	82.09±0.7
	Pagerank	80.51±0.6	80.19±0.6	**80.06±0.7**	80.99±0.8
	Random	82.23±0.7	82.00±0.6	82.10±0.6	82.24±0.6
Polblogs	Betweenness	**75.19±0.9**	77.88±3.0	76.41±1.8	83.17±2.5
	Closeness	75.87±1.4	79.09±2.4	75.72±1.8	83.02±2.0
	Degree	75.65±1.3	78.87±2.0	76.27±1.4	82.48±2.9
	Eigenvector	76.52±0.9	77.69±2.1	76.62±1.7	82.41±2.7
	Pagerank	75.25±1.4	78.09±2.2	75.99±1.6	82.73±2.6
	Random	79.93±3.3	80.25±2.9	80.13±3.4	83.57±3.4
Pubmed	Betweenness	84.93±0.3	84.71±0.2	84.38±0.2	85.21±0.3
	Closeness	85.42±0.3	85.38±0.2	85.24±0.3	85.58±0.3
	Degree	84.79±0.3	84.55±0.3	84.34±0.3	85.08±0.4
	Eigenvector	85.45±0.3	85.49±0.2	85.40±0.3	85.65±0.2
	Pagerank	84.54±0.4	84.20±0.3	**84.08±0.3**	85.13±0.2
	Random	85.83±0.3	85.74±0.3	85.64±0.3	85.90±0.3

B DETAILED EFFICIENCY RESULTS

We show the detailed runtime for Structack combinations in Table 8 and the memory consumption in Table 9. Runtime and memory consumption results are obtained after setting random linking with each selection method, and random selection with each linking method. We chose random because its time and memory requirements are negligible for our comparison. For these experiments, we also set the perturbation rate r to 0.05.

Table 8: Runtime in seconds for each selection/linking method.

	Dataset	Citeseer	Cora	Cora-ML	Polblogs	Pubmed
Centrality	Betweenness	170.42	242.02	398.90	233.17	19,507.80
	Closeness	36.95	54.19	91.24	66.79	3,938.98
	Degree	**0.18**	**0.42**	**0.46**	**0.96**	**2.82**
	Eigenvector	1.04	2.10	6.12	5.57	21.56
	Pagerank	2.01	2.22	3.09	4.65	20.86
Similarity	**Community**	**2.38**	**3.06**	**5.60**	**13.48**	**110.42**
	Distance	3.27	5.86	13.48	49.53	468.35
	Katz	58.75	79.47	85.60	38.30	5,978.16

Table 9: Memory consumption in Megabytes for each selection/linking method.

	Dataset	Citeseer	Cora	Cora-ML	Polblogs	Pubmed
Centrality	Betweenness	367	366	363	364	438
	Closeness	369	367	365	365	438
	Degree	**316**	**318**	**321**	**323**	**423**
	Eigenvector	357	354	353	361	437
	Pagerank	349	349	354	367	500
Similarity	**Community**	**368**	**367**	**370**	**387**	**1,055**
	Distance	387	382	400	409	2,150
	Katz	549	609	659	432	1,3881

C DETAILED UNNOTICEABILITY RESULTS

We show the unnoticeability of different combinations of Structack and with different perturbation rates $r \in \{0.001, 0.002, 0.003, 0.004, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.10, 0.15, 0.20\}$. These are the detailed results of what Figure 6 summarizes.

Notice that the choice of similarity has no impact on $r_{critical}$. For Citeseer, Cora and Cora-ML, closeness centrality and eigenvector centrality are the least noticeable. For the other two datasets (Polblogs and Pubmed), all centrality measures have the same $r_{critical}$

Table 10: Attack unnoticeability on each dataset after applying Structack with each centrality×similarity combination. The centrality measure (excpet random) with the highest critical perturbation rate $r_{critical}$ in each dataset is shown in boldface.

Dataset	Similarity Centrality	Community	Distance	Katz	Random
Citeseer	Betweenness	0.0100	0.0100	0.0100	0.0100
	Closeness	0.0250	0.0250	0.0250	0.0250
	Degree	0.0100	0.0100	0.0100	0.0100
	Eigenvector	0.0250	0.0250	0.0250	0.0250
	Pagerank	0.0100	0.0100	0.0100	0.0100
	Random	0.0500	0.0500	0.0500	0.0500
Cora	Betweenness	0.0100	0.0100	0.0100	0.0100
	Closeness	0.0250	0.0250	0.0250	0.0250
	Degree	0.0075	0.0075	0.0075	0.0075
	Eigenvector	0.0250	0.0250	0.0250	0.0250
	Pagerank	0.0075	0.0075	0.0075	0.0075
	Random	0.0250	0.0250	0.0250	0.0250
Cora-ML	Betweenness	0.0100	0.0100	0.0100	0.0100
	Closeness	0.0100	0.0100	0.0100	0.0100
	Degree	0.0050	0.0050	0.0050	0.0050
	Eigenvector	0.0100	0.0100	0.0100	0.0100
	Pagerank	0.0050	0.0050	0.0005	0.0050
	Random	0.0250	0.0250	0.0250	0.0250
Polblogs	**Betweenness**	0.0020	0.0020	0.0020	0.0020
	Closeness	0.0020	0.0020	0.0020	0.0020
	Degree	0.0020	0.0020	0.0020	0.0020
	Eigenvector	0.0020	0.0020	0.0020	0.0020
	Pagerank	0.0020	0.0020	0.0020	0.0020
	Random	0.0100	0.0100	0.0100	0.0100
Pubmed	**Betweenness**	0.0030	0.0030	0.0030	0.0030
	Closeness	0.0030	0.0030	0.0030	0.0030
	Degree	0.0030	0.0030	0.0030	0.0030
	Eigenvector	0.0030	0.0030	0.0030	0.0030
	Pagerank	0.0030	0.0030	0.0030	0.0030
	Random	0.0050	0.0050	0.0050	0.0050